A Conversational Case-Based Reasoning Help-Desk Utility for Complex Products

GIL CHEN and YORAM REICH
The Iby And Aladar Fleischman Faculty of Engineering
Tel Aviv University
ISRAEL
yoram@eng.tau.ac.il  http://www.eng.tau.ac.il/~yoram

Abstract: This research presents a conversational case-based reasoning (CBR) help desk utility: a system that provides service support to a company’s products. The goal is to utilize an existing database of customer calls as a case base, and achieve high recall and precision. In large case bases with inputs that may include irrelevant information, a conversation between the system and the user is required to gain more information about the problem. The suggested help desk utility addresses these concerns by using a 2-steps method. The first step is to create a category tree to enable quick and direct access to the problem’s general subject and characteristics. The second step is an interactive dialog with the user by asking relevant questions until a solution is found. The incremental clustering algorithm COBWEB was used to build the category tree and an information gain ratio was used to order potential questions. The system was evaluated by the technical support team at a CAD company and showed a practical potential.

Key-Words: intelligent front-end, hierarchical clustering, case-based reasoning, CAD

1. Introduction
In increasingly competitive markets, customers’ satisfaction is a vital corporate objective. Key elements to increasing customer satisfaction include producing consistently high quality products and providing high quality customer service. Many companies, particularly in the computer industry, provide telephone support for their products. Companies often provide a single national point of contact for customers, and may receive thousands of support requests daily. These telephone service centers, Help Desks (HD), or call centers, must be staffed by personnel who can provide the appropriate level of consistent support. Unfortunately, there are major problems that managers of HD encounter including lack of product-knowledgeable personnel with high staff turnover. Internet based applications that let the customer access a knowledge base of Frequently Asked Questions (FAQ) are also ineffective and insufficient.

Help desks and customer support are very successful applications of CBR (Case Based Reasoning) [1,2,3,5]. CBR is a method that uses earlier experiences or cases as the basis for decision-making. Figure 1 shows an enhanced cycle suggested in [8] to describe better the steps involved for building and maintaining the database as implemented in this research. This cycle includes a dynamic database-building step that can be used also for storing new cases and a maintenance step that manages the quality of the case-base.

![Figure 1: Enhanced 7-steps CBR cycle](image)

CBR provides methods for capturing new problem-solving experiences, so maintenance of the HD is inexpensive compared to other methods. This research suggests a technique that helps solving problems in domains with complex problems that require experience, time to think about the source of the problem, and in-house tests before arriving at a good solution. Examples of such HD are those dealing with operating commercial CAD (Computer-Aided-Design) software.

The order to address such problems, this research combines CBR with several other ideas:
Using CCBR (conversational CBR) [1] for solving a problem quickly and accurately.

Creating a classification tree or multiple indexing structure of cases [4, 6, 9, 10].

Using an entropy measure to select the “best” question to display to the user in order to eliminate irrelevant cases.

Using natural language processing to improve case match [1].

Section Error! Reference source not found. discusses related work. Section 2 describes the solution Section 3 discusses the evaluation of the method and Section 4 concludes.

2. Problem analysis and solution

2.1. Solution outline

Observing the difficulties in solving problems with complex representations including how professional solve them today, leads to several conclusions:

1. Problems could be classified in categories with some overlap in their descriptive attributes.
2. Problem categories could be organized in a hierarchal manner.
3. Users can relate a problem to a general category or environment, and give keywords that describe their problem.
4. Users cannot supply all information needed to understand the exact problem. They should be asked questions about the problem, until an accurate specification of the problem is defined.
5. The majority of the problems are repetitive or variants of such.

The proposed solution is based on these findings and includes three main steps:

- Automatic unsupervised category tree creation of all cases. This step is derived from conclusions #1, #2, and #5.
- Finding in the category tree the category most similar to the new problem. This step is derived from conclusions #2, #3, and #5.
- “Interactive dialog” with the user asking questions to reduce the number of candidate solutions. This step is derived from conclusions #4 and #5.

When the user introduces a new problem, the system traverses dynamic category tree to find the most appropriate category. When such category is found, the system presents the user with a list of potential cases that resemble the new problem. The user can select any of these cases to check whether the solution introduced in the case solves his/her new problem. The user may reduce the number of potential cases by answering questions that the system asks. In order to minimize the number of questions the user is asked, the system tries to answer some questions based on the selected category’s characteristics. When the problem is solved, it can be added to the case base as a solved case with all initial inputs the user gave plus all question/answer (Q/A) pairs that were asked and answered during the dialog between the system and the user. This procedure is an imitation of the technical support expert’s method of solving new problems: The expert “relates” a problem to a subject and then starts to ask questions that are relevant only to that subject in order to reduce possibilities and shorten the time to solve the problem.

2.2. Case collection and analysis

252 cases were used to create the initial case base. These were collected from the following sources:

- The database of the technical support department of a SolidWorks dealer (Systematics LTD). SolidWorks®, was chosen as the CAD application for this study, but the method will be similar for other CAD products.
- Official on-line help documentation.
- SolidWorks web site knowledge base (FAQ and technical tips & tricks)
- SolidWorks newsgroup (comp.cad.solidworks)
- Personal eight years of experience in CAD technical support.

2.3. Problem (Case) Representation

The representation of a problem in a CAD application is much complex than in many well-known domains such as a printer-technical-support. Two random problems in the domain may have only few attributes in common. They both have name, description, date, version of application used, operating system (OS), hardware platform, drivers’ information, system’s settings etc. Asking user to provide all these attributes is useless. He/she will give up and will not spend time looking for them. Another reason not to represent a problem with all these attributes is that many problems occur (and solved) regardless of many of them. For example, in most cases, a failure in generating a complicated fillet feature is caused by its wrong definition and is independent of the OS, hardware, or drivers.

Three guidelines led to the case representation:

- Start with few essential attributes and ask more information as the process advances.
- Keep it simple. The target market we focus on includes novice users who may not be able to supply all information at once.
- All additional information given to the system during the “dialog” step and the solution itself should be automatically added to the representation of the case.

The representation of the case includes the following attributes:
Case ID: A unique number to identify a case.
Case name: A name of the problem, such as “New column in BOM.”
Case Description: More accurate and detailed description of the problem. For example: “A new column I added to the BOM is not displayed when the BOM is inserted to the drawing.”
Date
Software Version and Service Pack
Keywords: Words that describe the problem including verbs, adjectives, and nouns, such as export, DXF, layer, or BOM. A user may define as many keywords as he/she likes.
Environments: There are three working environments in SolidWorks: Part, Assembly, and Drawing. Problems may be general ones, such as installation problems. The forth “environment” is a “General,” which deals with all these kinds of problems. A case may be represented by a combination of environments. For example, a Bill Of Material problem may be defined as a Drawing related problem and/or an assembly one.
Topics: The Topics list was taken from SolidWorks Knowledge Base. This list is quite similar to the first level headers of the online help and reflects the various subjects in the application. A topic may be Data Exchange, Detailing, Features, Mates, Dimensions, Notes, etc. A case may have multiple topics. For example, a dimension problem of an imported DWG file may be associated to Dimensions and Data Exchange.
Dialog Questions/Answers: All questions that were asked during the conversation with the user and answers the user gave. The questions are phrased in a way that there are a list of answers the user should select from. For example, “Is it an assembly related problem?” which has only two possible answers (“Yes,” “No”) or “What file format is required?” which has many possible answers such as “IGES,” “STEP,” or “DXF.”
Solution (Actions): This includes all steps to solve the problem, whether a single or multiple action.

2.4. The algorithm
The category tree is created with the incremental unsupervised conceptual clustering system ECOBWEB [7]. This category tree is used as an indexing mechanism for cases. Retrieval is not based on a traditional similarity measure between cases and the new problem but is done incrementally by traversing the hierarchy from its root to its leaf or internal categories.

If the category that was selected in the previous step includes few cases, the user may check their solutions and see whether one solves his problem. This is rarely the case; therefore, a dialog is formed for focusing on relevant cases including two steps:
1. Initial Category selection.
2. Question selection mechanism.

After the user approves the proposed category (or rejects it and select a different one), the system advances to the next sub step to conduct the dialog with the user. At this stage, all Q/A pairs of all the cases of the selected category should be analyzed and only the “good” ones should be displayed in an importance order (see Figure 2). We define a “good” question as a question that distinguishes the most between cases.

### OrderQuestions()

1. Collect all Q/A pairs from all cases in the selected category to PotentialCasesList and remove repetitive Q/A and questions of accepted characteristics from it.
2. Compute GainR for all Q/A pairs in PotentialCasesList
3. Add Q/A pair with highest GainR to QuestionList
4. Do until (QuestionList covers PotentialCasesList)
   1. Set NotCoveredYet = (all cases in PotentialCasesList that are not covered yet with QuestionList)
   2. Compute GainR for all Q/A pairs in NotCoveredYet
   3. Add Q/A pair with highest GainR (and not already in QuestionList) to QuestionList
5. Show QuestionList to the user

PotentialCasesList is a list of all potential cases that may have the solution to the new problem,

QuestionList is a list of all questions that should be asked (ordered by GainR),
(QuestionList covers PotentialCasesList) is a functions that returns True when the question list in QuestionList covers all cases in PotentialCasesList, and

NotCoveredYet is a list of all potential cases that are not covered yet with QuestionList

Figure 2: Question Ordering Algorithm

All Q/A pairs are ordered according to their Information Gain Ratio, so the optimal action is to answer the first question in the list. If the user decides not to answer the first question, he should select the second “good” question, the third, etc. This may happen if the user decides that the question is irrelevant or because the user does not know the answer. The system finds and displays only the $N$ first questions that cover all the potential cases.

After the user answers a selected question, the system removes all cases that do not have this Q/A pair as one of the attributes from the set PotentialCasesList. This operation reduces the number of potential cases. This procedure is repeated until the user finds a solution. If all of the potential cases do not solve the problem, the user can select another parent category and start this process again.
Before starting to compute the GainR values, all Q/A pairs are grouped and a removal process is applied to remove all repetitive Q/A pairs and questions that appear as approved Q/A characteristics from this group.

Once this algorithm is used and the user selects a question and answers it, the system applies the same algorithm once more, this time without the first step of collecting and arranging the potential case list. Instead, it removes all cases from PotentialCasesList that don’t have the Q/A combination, leaving only relevant cases in the list for further analysis.

3. Results

TeSAS (Technical Support Aiding System) is a Windows-based system with Graphical User Interface (GUI) written in Microsoft ACCESS® database and application. We selected to test TeSAS only with human experts. In the evaluation, expert users were testing it and commenting on it. They input real problems from the company’s database and tried to solve those using TeSAS.

The evaluation process had 5 stages:
1. Analyzing a real world case base of a CAD company’s technical support department to focus on most common subjects and to collect hundreds of previously solved real problems.
2. Filling in a keywords questionnaire by the technical support team members.
3. Evaluation of TeSAS by the technical support people.
4. Analyzing evaluation results and testers feedback and enhancement suggestions.
5. Conducting additional studies to enhance the application based on findings and feedback.

3.1. Stage 2: Keyword Questionnaire

We asked seven members of the technical support team to fill in a questionnaire about keywords of cases. This stage is important because when users search for cases, they input keywords. If different users input completely different keywords, it would be impossible to build an application. Testers were given a set of eleven problem descriptions and had to fill in keywords that best describe the problem. They were guided to select keywords as if they were regular users. These keywords were also analyzed by a synonym mechanism. Table 1 summarizes the results. Although only 38% of the user’s input is identical to the keywords of the relevant case in the case base, it improves to 62% after the synonym mechanism. This proved to be a very important step in the process of focusing on relevant categories in the second algorithm (interactive questions).

It turns out that keywords play a role in defining the difficulty of problems. In the subsequent evaluation, expert were given problems at three difficulty levels: easy, medium, and complex problems. An “easy” problem is one that is explicitly defined, keywords can be easily chosen (and most of them will be identical to the keywords in the case base). The problem description is very simple yet accurately describes the problem. An easy problem has one (or few) solution. An example of such problem is: “How can I convert a part’s sketch to Autocad®?” Potential keywords may include: Autocad, convert, or sketch. The synonym mechanism might change: [Autocad→DXF, DWG], [Convert→export]. This should be sufficient information to solve the problem (create a drawing of the sketch and use File→Save As→Dxf or Dwg).

<table>
<thead>
<tr>
<th>Table 1: Keywords Selection</th>
<th>In case base</th>
<th>As defined by the users</th>
<th>After synonym mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average keywords in a case</td>
<td>5.8</td>
<td>5.0</td>
<td>4.7</td>
</tr>
<tr>
<td>Standard deviation of # of</td>
<td>1.7</td>
<td>1.0</td>
<td>1.2</td>
</tr>
<tr>
<td>Average % of identical</td>
<td>38%</td>
<td>62%</td>
<td></td>
</tr>
<tr>
<td>keywords in a case</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard deviation of</td>
<td>0.19</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>identical keywords</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A complex problem might be: “Why can’t I open a Parasolid file created with SolidWorks in another application?” Selecting keywords for this problem is not trivial. Potential keywords may include: Parasolid, can’t, open, other, application. The only useful keyword here is Parasolid; other words may be useless or misleading. The word “open” implies an import problem, but actually, it is an export one, and the system may fail to focus on the right category. This problem may have many reasons and many questions should be asked during the process until the right solution is found.

In order to check the definitions for simple, average, and complex problems, a questionnaire was given to the technical support team with simple, average, and complex problem descriptions. Seven experts participated in the experiment. They each had to provide keywords to 6 simple problem cases, 3 average problem cases, and 2 complex problem cases. They filled in the keywords they think were the most appropriate. For each case, the percent of overlap between the keywords given by the participant and the case keywords were calculated.

The results are graphed in Figure 3. It shows the average and standard deviation, maximum, and minimum overlap in the keywords given for each of the three problem categories. The graphs show clear separation between the three groups, providing an operational definition for the three groups of
problems. In order to test that the three groups are indeed different, we used a $t$-test without assuming equal variance and with a conservative degree of freedom setup $dof = \min(n_1 - 1, n_2 - 1)$, where $n_1$ and $n_2$ are the sizes of the two samples. The results show that the simple and medium-complexity samples are different at significance level 0.01 and so are the average and complex samples. This analysis provides the necessary basis for treating the problem classes as different in the rest of this study.

The results of the evaluation are presented in Table 2. Each entry includes the average and the standard deviation in parenthesis. While the number of testers and questions per tester are not large, therefore statistical analysis is difficult to perform, the differences between the question categories, and the utility of the approach in assisting users are clearly demonstrated.

### Table 2: Evaluation Results

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Medium</th>
<th>Complex</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of problems for each of the five testers</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>1. % problems that were solved</td>
<td>93% (9%)</td>
<td>60% (15%)</td>
<td>30% (27%)</td>
<td>73% (32%)</td>
</tr>
<tr>
<td>2. % Q/A assumptions that were approved by the user</td>
<td>88% (11%)</td>
<td>83% (14%)</td>
<td>20% (5%)</td>
<td>82% (34%)</td>
</tr>
<tr>
<td>3. Average number of questions correctly answered by the system</td>
<td>2.2 (0.3)</td>
<td>1.3 (0.1)</td>
<td>0.5 (0)</td>
<td>1.6 (1.2)</td>
</tr>
<tr>
<td>4. % times questions were relevant and ranked 1-3 in the interactive questions list</td>
<td>75% (39%)</td>
<td>23% (25%)</td>
<td>0%</td>
<td>47% (45%)</td>
</tr>
<tr>
<td>5. % problems solved after 1 question</td>
<td>52% (14%)</td>
<td>20% (14%)</td>
<td>0%</td>
<td>35% (19%)</td>
</tr>
<tr>
<td>6. % problems solved after 2 questions</td>
<td>41% (13%)</td>
<td>40% (15%)</td>
<td>0%</td>
<td>33% (26%)</td>
</tr>
<tr>
<td>7. % problems solved after 3 questions</td>
<td>0%</td>
<td>0%</td>
<td>30% (27%)</td>
<td>5% (21%)</td>
</tr>
</tbody>
</table>

The maximum percent was 100%, the large STD is due to the skewed variability.

The most salient feature is the difference between simple and medium-complexity problems and complex ones. While simple and medium-complexity problems could be solved in high percentage (93% and 67% respectively), only 40% of the complex problems could be solved using TeSAS. The duration of conversation is short when a simple or medium-complexity problems are introduced. This is mainly due to the exact category selection while searching the best branch in the category tree. This yields good Q/A assumptions and very few additional questions are required to focus on the exact problem and solution.

When the problem is complex, searching the tree may find the wrong category, with wrong assumptions and irrelevant questions. This forces the user to climb (at least) one level upward, reaching more general category, with more general assumptions that might be wrong and might display irrelevant questions.

Another interesting point is the total number of questions asked until a solution is found. When the problem is simple, more than 2 questions are asked and automatically answered by the system (assumptions) plus another $\sim1.5$ during the conversation. With medium-complexity problems, the number of correct assumptions is 1.3 while 2
more questions should be asked further. With complex problems, only 0.5 questions are correctly assumed while there are 3 more additional questions to be asked to find a solution. This looks reasonable: When the problem is simple and a solution may be found almost automatically, the most appropriate category is found, many questions may be automatically answered, and only few are required to solve the problem. As the problem becomes more complex, finding the right category becomes more difficult, fewer questions can be correctly assumed and a longer conversation with the user is required.

4. Discussion and Conclusions

This work proposed a method for building a case base to aid users solve technical problems in a complex domain with many different problems described by many attributes. The main concept is to combine conversational CBR techniques, clustering algorithm, decision-tree creation method, and knowledge acquisition concepts. The COBWEB clustering provides a flexible, adaptable case organization and retrieval mechanism. This is highly desirable when dealing with domains that evolve constantly, such as software products. COBWEB provides another key benefit to CCBR. When a new case traverses the hierarchy to find past relevant cases, the characteristics of visited categories create assumptions about questions that are relieved from the user. The user can subsequently approve or disprove some of them and TeSAS continues accordingly. This mechanism can be seen as an alternative to other approaches to minimize questions such as taxonomies and causal relations or decision forests. The benefit here is that no effort is spent to create specialized representations beyond the retrieval structure that is built automatically from cases.

Another novel part of this work is the characterization of problems into three subclasses based on experts’ ability to characterize them (and not necessarily solve them); the subclasses are easy, medium, and complex problems. It has been shown that the difficulty of CCBR is in solving complex problems and that this difficulty is due to the difficulty to characterize them accurately.

The proposed method is applied on SolidWorks® customers queries case base, but may be implemented on every other CAD application and probably on every other application and complex product. Tests with technical support team experts show that the present approach can address simple and medium-complexity problems reasonably, but performs mediocrity in complex cases. In most cases, the majority of problems encountered by users would be easy or medium complexity problems. Therefore, the approach as is could be used as a help-desk utility deployed over the Internet.

Acknowledgements

We would like to express our sincere thanks to Systematics LTD for helping us to collect the problems and participate in evaluating the system.

References